ソフトウェア工学　52－16
（１９８７・２・１４）

ペトリネットを利用した並行システムの動作解析

孫　立寧　　落水　浩一郎
（　静岡大学　工学部　情報工学科　）

本報告は複数の動作実体がリンクを介して、相互作用するような並行システムの動作解析系について述べる。　本解析系では、実体（プロセス、モジュール等）の動作記述を拡張状態遷移図で記述し、リンクの構造とその特性を指定した上で、全体をまとめてペトリネット表現に変換、解析し、解析結果を再び拡張状態遷移図、及びリンク表現に逆変換する。　すなわち、ユーザとのインタフェースに状態遷移図を用い、ペトリネットによる解析の詳細を解析系の内部に隠す。　最後に実際応用における本解析系の解析手順を、HDLCの解析を通じて検討する。　現在、VAX11-780 上のFranz-lispで実現したプロトタイプができている。

Analysing Behaviour of Concurrent Systems Based on Petri-net

Li-ning SUN　　Koichiro OCHIMIZU
Dep.of Computer and Information Science,Faculty of Eng.,SHIZUOKA UNIV.,3-5-1 JYOHOKU,HAMAHATSU 432,JAPAN

This paper describes an analysis system to validate the behaviour of concurrent systems. In the system, we represent objects(process,module) as ETDs (extended transition diagrams) and their interactions as links. Analysis is performed as follows; (1) Designate ETDs and links. (2) Each of them is converted into Petri-net, and all of them are combined into a unified Petri-net. (3) Re-convert results of analysis into ETDs or link representation. We use ETDs as man-machine interface and hide the details of analysis based on Petri-net in the system. An example of HDLC analysis is given and the effectiveness and usefulnesss of the system is discussed.The prototype system was implemented in Franz-lisp on VAX11-780.

1．はじめに

設計段階にあるソフトウェア・システムの動作を解析することは、誤りの早期発見、より良い設計代替案の採用等に有効である。

このためには、システムの動作分析ができるように、特定の観点からシステムの動的振舞いを表現する記述体系が必要であり、例えば、状態遷移図[1]、ペトリネット[2]などが使用されている。 この中で、従来から使われてきた状態遷移図は、外部信号による状態遷移、及び各状態の果たす制御全体における役割が明瞭に記述できるが、相互作用が複雑な並行システムにおいては、系統的解析方法はない。並列処理システムのモデル化に用いられたペトリネット表現は、ネットワーク・プロトコルの検証をはじめ[3]、解析的な面への応用が広がりつつある[4]。ペトリネットはそのモデル化能力が高く、解析モデルとしてもっとも基本的なものであると考えられるが、状態遷移図と比べると、構造が複雑、記述量が多い等の欠点を持つので、大規模ソフトウェアの開発には応用しにくいと考えられる。

我々は複数の動作実体が、リンクを介して相互作用するような対象の動作解析のための解析系の開発を以下のような観点からすすめている。 図1において、実体内部の動作表現を状態遷移図であらわし、リンクの構造とその特性を指定した上で、全体をまとめて、ペトリネット表現に変換、解析し、解析結果を再び状態遷移図表現、及びリンクの表現に逆変換する。 すなわち、ユーザとのインタフェースは状態遷移図を使い、解析の詳細は解析系の内部に隠すという方針である。 解析はペトリネットの性質のうち、Boundedness、Liveness、Home-state、Proper-terminationを使用する。

第2－6章では、資源管理問題を例にとり、拡張状態遷移図、ペトリネットへの変換、リンク、結合、解析、及び逆変換等の、解析系の基本技法を説明する。 第7章では、HDLCの解析を通じて、実際応用における本解析系の解析手順を検討する。 第8章では、まとめと今後の課題を述べる。

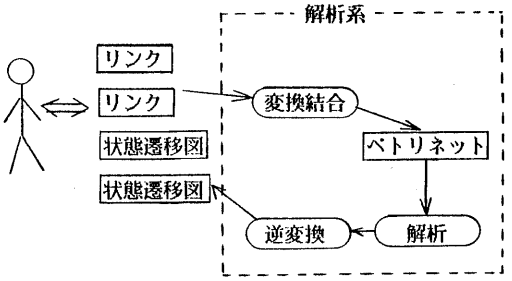


図1． 解析系の概要

2．拡張状態遷移図とペトリネットへの変換

2.1 拡張状態遷移図 拡張状態遷移図はソフトウェア・システムを構成する実体集合（プロセス、モジュール等）の動作を記述するために、状態遷移図に動作遅延時間の概念を導入したものである。 拡張状態遷移図では、ソフトウェア・システムの実現予定コード部分を、表現目的や分析目的により、有限個の状態で抽象する。 各状態における実際の処理時間をその状態における遅延時間と考える、すなわち、図2において、事象aの発生後、事象bを受け付けるまでには、t時間単位必要とするという意味である。 このとき、遅延時間中に発生した入力は発生順に記憶できることとし、また、入出力としては基本事象の論理積表現も許すものとする。

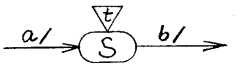


図2． 遅延時間

定義1： 拡張態遷移図は5項組（$Q\times\tau$，Σ，Δ，δ，Γ，τ）である。

$Q\times\tau$： 状態の有限集合、τが遅延時間、自然数。$\tau=0$のとき、単にQと略記することもある。

Σ： 入力の有限集合

Δ： 出力の有限集合

δ： $Q\times(\tau>0)\rightarrow Q\times(\tau-1)$

遅延時間の経過、または

$Q\times(\tau=0)\times(\cap\Sigma)\rightarrow Q\times\tau$

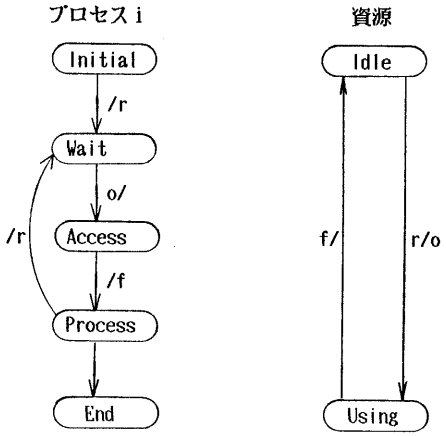
現在の状態と現在の入力を次の状態へ写像する

状態推移関数の有限集合

Γ： Q×（t＝0）×（∩Σ）−>∩Δ

現在の状態と現在の入力を出力へ写像する出力関数の有限集合

図3に複数のプロセスが共通の資源をアクセスする問題の拡張状態遷移図による記述を示す。 三角形の中の数字は遅延時間を表す。



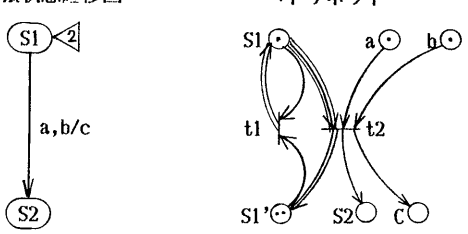
r：資源要求の発生

o：受理

f：資源解放の要求

図3．拡張状態遷移表現による資源管理問題

2.2 ペトリネットへの変換 拡張状態遷移図をペトリネットで表現するためには、遅延時間の表現できるペトリネットが必要となる。 文献[5]では遅延時間をトランジションの発火に要する時間であるとして、時間をトランジションに割り付けるのに対して、文献[6]では遅延時間をプレースの経過に要する時間であるとして、時間をプレースに割り付ける。これらの提案はペトリネットの記述要素を増加させ、また発火規則を複雑にし、その結果、解析力を低下させる。 我々は拡張状態遷移図の特性から、通常のペトリネットで、遅延時間を図4のようにトークン数により表現する、すなわち、S1において、トランジションt1を二回発火してから、入力a，bを受け付けるトランジションt2を発火できる状態となる。



S1,S2∈S、S1'∈R、a,b,c∈D

t1,t2∈T

I(t1)＝{S1,S1'} O(t1)＝{2S1}

I(t2)＝{3S1,a,b} O(t2)＝{S2,c,2S1'}

初期ﾏｰｷﾝｸﾞ：{S1,a,b,2S1'}

図4．ペトリネットへの変換ルール

定義2： 拡張状態遷移図（Q×τ，Σ，Δ，δ，Γ）に対して、ペトリネット（P，T，I，O）を次のように対応させる。

Pはプレースの有限集合であり

P＝S∪R∪D。

ここで、Sは状態プレースの有限集合であり

Q×τに対応させ、

Rは時間プレースの有限集合であり

Q×τ（τ≠0）に対応させ、

Dはデータ・プレースの有限集合であり

Σ∪Δに対応させて定義する。

このとき、トランジションの有限集合Tは

$T=\{ts,r \mid s\in S, r\in R\} \cup \{ts,d \mid s\in S, d\in D\}$

のように定義する。 ここで、ts,r は遅延の経過に伴う遷移を表し、ts,d は入力事象の発生による状態遷移を表す。 トランジションtを入力／出力プレースの多重集合へ写像する入力／出力関数をそれぞれ

$I(t)=\{((\tau+1)*s,d) \mid ts,d\in T\} \cup \{(s,r) \mid ts,r\in T\}$

$O(t)=\{(\delta(s,d),\Gamma(s,d),\tau*r) \mid ts,d\in T\} \cup \{(2*s) \mid ts,r\in T\}$

のように定義する。

プレース中のトークン数は以下のように解釈する。

- 時間プレース中のトークン数は残遅延時間を表し、初期マーキングでは、遅延時間に設定される。
- 状態プレース中のトークンは拡張状態遷移図の現状態に対応する。 状態プレースにn（n＞1）個のトークンがあることは、その状態での計算がすでにn－1単位時間を経過したことを表す。
- データ・プレース中のトークンは入出力事象の発生を表す。

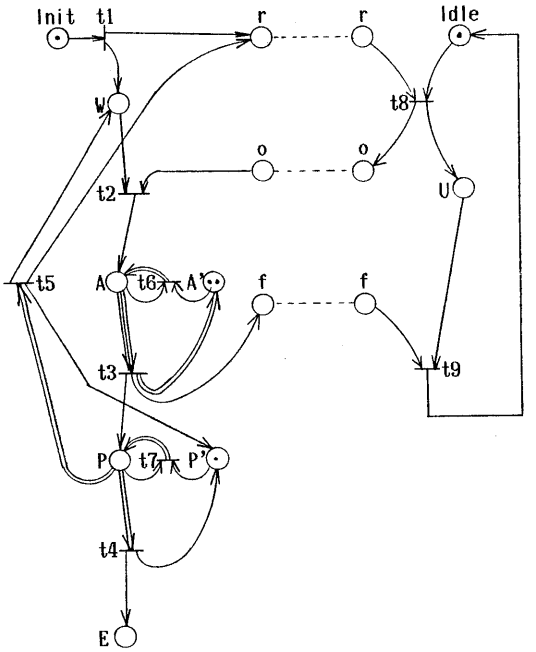


図5．ペトリネット表現による資源管理問題

図5は図3資源管理問題のペトリネット表現である。図5において、ある実体で閉じない入出力事象（r，o，f）が存在する。 これは他の実体に伝える／から伝えられる事象を表し、外部事象と呼ぶことにする。

3．リンクと結合

3.1 リンク

実体間のメーセッジ交換はリンクを介して行う。 リンクはバッファ、キューを抽象したものであり、容量、通信手段等の特性を指定できる。 本解析系では、汎用リンクを利用者の指定によって、インスタンシエトするツール群を準備している。

3.1.1 バッファ

バッファは同種類のトークンを転送するリンクであり、有限容量、または、無限容量のものを指定できる。

定義3.1 ： 有限容量のバッファは次のように定義する（図6(a)）。

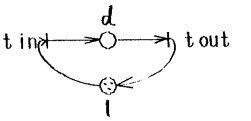
P＝{d，1} ここで、プレースdはトークンを貯める役割を果たし、リンク・プレースと呼び、プレース1中のトークン数はバッファの残容量を表し、容量プレースと呼ぶことにする。

T＝{tin，tout}

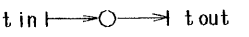
I（tin）＝{1}　　O（tin）＝{d}

I（tout）＝{d}　　O（tout）＝{1}

定義3.2 ： 容量プレース中のトークン数が無限のとき、無限容量バッファという。 このとき、容量プレースは記述しない（図6(b)）。



(a) 容量3のバッファ



(b) 無限容量のバッファ

図6．バッファ

3.1.2 キュー

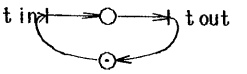
キューは異種類のトークンをFirst-in-First-outで記憶・転送するリンクである。 容量と転送するトークン型を指定できる。

定義4.1 ： 容量が1、トークン型の種類が1、のキューは容量1のバッファとし、単位キューと呼び、Kで表す（図7(a)）。

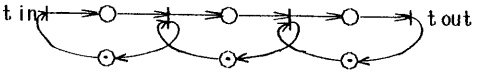
定義4.2 ： 容量がn、トークン型の種類が1、のキューはn個の単位キュー（K1,K2,…，Kn）を直列につなぎあわせる（1≦i≦n－1に対して、Kiのトランジションtoutと、Ki+1のトランジションtinと重ね合わせる）ことより構成されたものとし、nキュ

ーと呼ぶ（図7(b)）。

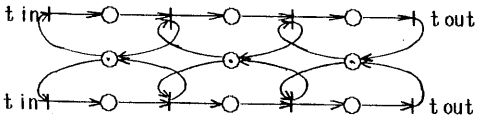
定義4.3 ： 容量がn、トークン型の種類がm、のキューはm個のnキュー｛（K11, …, K1n），…，（Km1，…，Kmn）｝を並列につなぎあわせる（1≦i≦n、1≦j≦m－1に対して、Kijの容量プレースとKij+1の容量プレースを重ね合わせる）ことより、構成されたものとし、m×nキューと呼ぶ（図7(c)）。



(a) 単位キュー



(b) 3キュー



(c) 2×3キュー

図7．キュー

m×nキューは、容量がnであり、各キューに種類毎にトークンを管理するとともに、全体の入力順序を保持するキューである。

3.2 結合　我々の解析手順は図8に示すように、拡張状態遷移図、リンクの各々をペトリネットに変換した上で、合成して一つの複合ペトリネットを作り上げ、複合ペトリネットの解析結果をもとの表記法に逆変換しようとするものである。

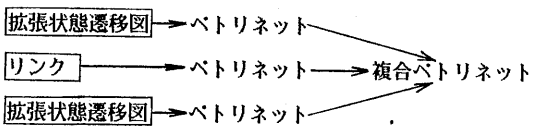
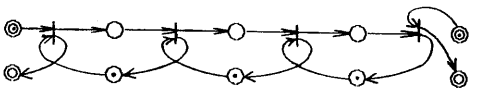
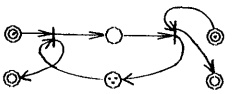


図8．複合ペトリネット

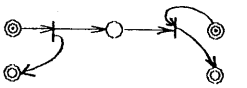
拡張状態遷移図間の交信は、おのおのの外部事象がどのような特性をもつリンクを介して、どのように結合されるかにより、全体の振舞いもまた変る。



(a) 容量3のBlockingキュー



(b) 容量3Blockingバッファ



(c) Noblocking バッファ

図9．結合（二重丸◎は動作実体のプレースを表す）

リンクと送信実体との結合法としては、基本的には2種類考えられる（図9）。

・Blocking：通信路バッファに空領域があるときに限って、実体はトークンの転送ができる。

・Noblocking：通信路バッファの状況によらずに、実体はトークンの転送ができる。

定義5： 結合とはリンクの特性と結合法の指定により、複数のペトリネットを一つのペトリネットにまとめあげることである。 まとめあげられた一つのペトリネットを複合ペトリネットと呼ぶ。

図5において、実体プロセスと資源とは、3個の無限容量のバッファr、o、fを介して、Noblocking結合法より結合する。

4．ペトリネットの解析

4.1 解析手法　複合ペトリネットそのものは単に一つのペトリネットなので、今まで研究されたいくつかの解析方法はそのまま使える。 本解析系では、文献[2] による可達木を以下のように修正して、解析する。

本来の可達木は初期マーキングから、ペトリネットの発火ルールに従い、可達可能なマーキング集合を木の形式で表現するものであり、もし、ある時点で、一つ以上のトランジションが発火条件を満たすとき、おのおののトランジションを発火させ新しいマーキングを得る。ところが、そ

れはペトリネットの解析に対して指数オーダのメモリ領域と時間を要するので、実際的な応用には使いにくい。 本解析系では、実体は他の実体との同期をとる部分以外は、並行に独立動作するという性質から、同時に発火可能な（競合関係にない）トランジションは同時に発火させるような変形の可達木（以後は簡単に可達木と呼ぶ）を求め、解析の効率を上昇させる。

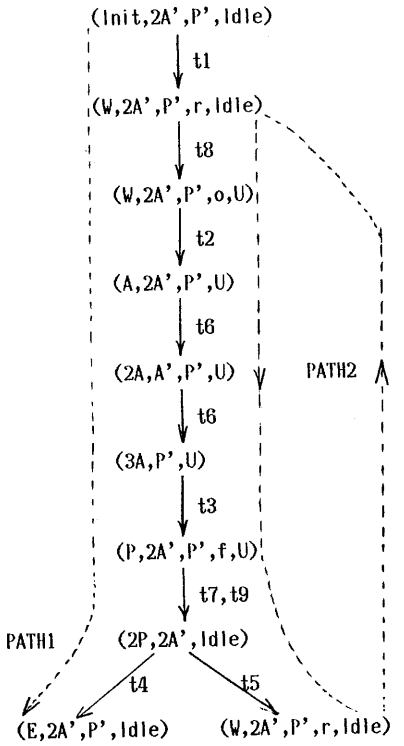


図10. 資源管理問題の可達木

図10に資源管理問題の可達木を示す。 マーキング(P,2A',P',f,U)では、t7とt9を同時に発火させ、マーキング(2P,2A',Idle) では、t4とt5に分岐する。

このように求めた可達木において、以下のような用語を定義する。

- 初期マーキングから可達可能なマーキングの集合をCで表し、そのうち、どのトランジションも発火できない（例(E,2A',P',Idle)）マーキングの集合を終端マーキングと呼び、Ctで表す。
- マーキングMから、マーキングM'に可達可能なら、MとM'間に可達パスがあるという。 もし、Mが初期マーキング、M'∈Ctなら、そのパスを切断パス（例Path1）といい、もし、M'=Mなら、そのパスを連結パス（例Path2）という。

4.2 ペトリネットの性質 可達木を利用してペトリネットにおける数多くの性質が解析できる。 状態遷移図の動作解析においては、次に述べるいくつかのペトリネットの性質が有効と考えられる。

性質1 Boundedness： もし、任意のM∈Cにおいては、すべてのプレースに含まれるトークン数が有限なら、ペトリネットはBoundedness という。

性質2 Liveness： もし、任意のM∈Cにおいて、任意のトランジションt∈Tが一定のトランジション・シーケンスの発火より、発火可能なら、ペトリネットはLivenessという。

性質3 Home-state： もし、あるM∈Cがすべての連結パスに含まれるなら、MがHome-stateといい、ペトリネットはHome-stateをもつという。

性質4 Proper-termination： もし、任意の切断パスの最終マーキングM∈Ctにおいて、すべてのデータ・プレース、及びリンク・プレースにトークン数が0なら、ペトリネットはProper-terminationという。

これらの性質の拡張状態遷移図、及びリンクの動作上における解釈は次のようになる。

- Boundedness が成り立つと、ペトリネットでは特にデータ・プレース、及びリンク・プレース上のトークン数が有限であることが保証される。 すなわち、リンクに留まりうる事象の数は有限である。
- Livenessが成り立つと、ペトリネットでは発火しないトランジションがないので、拡張状態遷移図で定義したすべての遷移は論理的に起こりうる。
- Home-stateを持つと、システムにおいて、ある特定の状態が再現されることを意味する。 そのようないくつかのシステムの動作間に共通の状態があることも意味する。

・Proper-terminationが成り立つと、拡張状態遷移図で定義した終了状態において、リンク中に受け付けを待つ事象はない。

5．解析手順

本解析系を利用した解析手順は以下のようになる。

(1) ソフトウェア・システムの動作における解析したい点、解析の仕様を決める。

(2) 性質1－4を使って、その解析の実現法を設計する。Boundedness はバッファ容量の設計、Livenessはソフトウェア中の非活性部分の検出、Home-stateはソフトウェアの特定の動作状態の確認、Proper-terminationは終了状態の確認に使えると考えられる。

(3) 解析結果を分析する。

解析結果は性質1－4が成り立つか、成り立たないかとなるが、後者の成立たない場合において、重要なのはその原因に関する情報を簡潔明瞭に逆変換機能を通してユーザに提供することである。 本解析系では、逆変換機能にわたす解析結果を次のように決める。

・性質1において、もし、Boundedness が成立たない場合には、そのマーキングに到達するパスを表示する。

・性質2において、もし、Livenessが成立たない場合には、そのトランジションを表示する。

・性質3において、もし、Home-stateが存在するなら、それを表示する。 または、利用者の指定のHome-stateを入力させ、これが含まらない連結パスを表示する。

・性質4において、もし、Proper-terminationが成立たない場合には、その切断パスを表示する。

6．解析逆変換

解析の逆変換では上記したペトリネットの性質の解析結果（マーキング、パス、トランジション）をもとの拡張状態遷移図表現に逆変換する。 図11に資源管理問題の切断パスの拡張状態遷移図表現を示す。

7．例

我々の提案した手法のソフトウェア・システム開発における適用範囲、及び実用性については、その手法を実際の

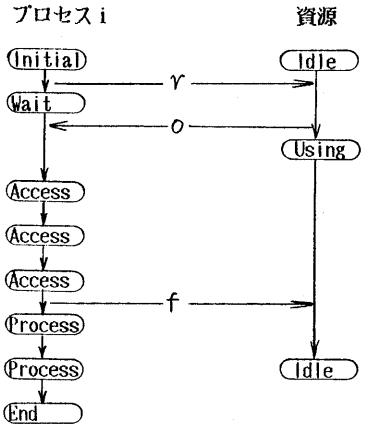


図11．資源管理問題の切断パス

ソフトウェア・システム開発に適用しつつ、その精度を高めていくことになる。 前例で挙げた資源管理問題のような並行処理に対しては、一段高いレベルの使いやすい解析手法を与えたと考えている。現在、VAX11-780 上のFranz-lispで実現したプロトタイプを完成し、いくつかの実例を解析してみた。 本稿では、DCNAにおけるHDLC手順を使用するデータ・リンク制御手順、不平衡型手順クラスＵＮの一部についての解析結果を述べる。

クラスＵＮの一部に関する一次局と二次局の動作を拡張状態遷移表で最初に表現したものを図12に示す。

一次局と二次局は、Noblockingバッファを介して、通信を行うとする。

(1) 解析の仕様

ａ．データ転送用のバッファがオバー・フローにならないか。

ｂ．終了モードによる以外は、行きつまることはないか。

ｃ．切断パスが正しいか。

ｄ．一次局と二次局間での情報フレーム通信における順序が正しいか。

ｅ．状態遷移図の中のすべての状態と遷移が論理的に必要であるか。

(2) 解析の実現法

解析の仕様をもとに、ａはBoundedness 、ｂ、ｃはProper-termination、ｄはHome-state、ｅはLivenessを使

って、解析の実現法を設計する。

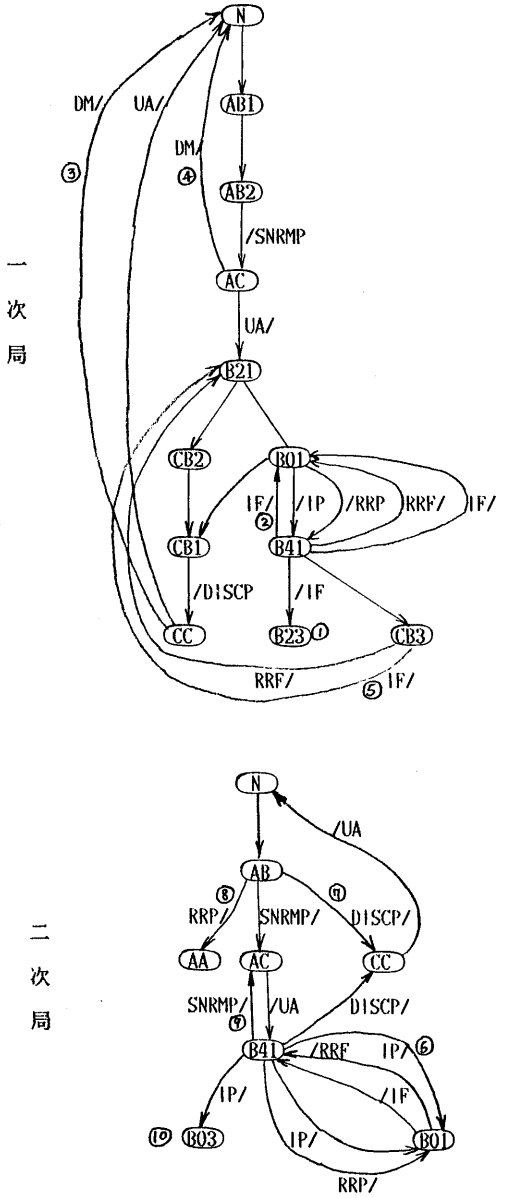


図12. クラスＵＮの一部の動作

(3) 解析結果の分析

本解析系を利用して解析し、解析結果を分析することにより、図12に存在した設計誤りを検出訂正した。

・Proper-terminationにより、状態1,10を削除、トランジション2,5,6 を追加。

・Livenessにより、トランジション3,4,7,8,9 を削除。

・Home-stateにより、トランジション2,6 を削除。

8．おわりに

本報告は拡張状態遷移図群とその間のリンクの特性をそれぞれペトリネットに変換、結合、解析、及び逆変換に基づく並行システムの動作解析手法について述べた。

本手法では、閉じたペトリネットの世界での解析を行うので、ペトリネット理論をそのまま、きれいに展開でき、しかも、今後ペトリネット理論の発展によって、より高いレベルの解析が期待できるものと思われる。

我々の提案した手法は、あくまでもソフトウェア・システムの動作解析作業を支援する道具にすぎないと考えている、　言い換えれば、Whatは教えてくれるが、Why は利用者の分析によるしかはない。

本手法を実際のソフトウェア開発への応用するにあたっては、

・グラフィックスのユーザ・インタフェース

・パフォーマンス評価方法

の二点を補充する必要であり、今後の課題である。

参考文献：

[1] 日本電電公社　"DCNA データリンクレベルプロトコル",財団法人日本データ 通信会

[2] J.L.ピーターズン,　"ペトリネット入門" , 共立,S59.

[3] P.AZEMA "Design And Verification of Communication Procedures: a Bottom-up Approach", In Proc. 3rd.int.Conf.,Software Eng.,May 1978

[4] T.AGERWALA "Some Applications of Petri Nets", In Proc. Nat.Electron.Conf.,Vol.32.Nat.Eng. Consortim,oct.1978

[5] C.V.RAMAMOORTHY "Performance Evaluation of Asynchronous Concurrent System Using Petri Net",IEEE Tran.Soft.Eng.Vol.SE-6,NO.5,Sep.1980

[6] J.E.COOLAHAN "Timing Requirements for Time-Driven Systems Using Augmented Petri Nets",IEEE Tran.Soft.Eng.Vol.SE-9,NO.5,Sep 1983